pca

October 12, 2023

1. Đọc thư viện và khảo sát dữ liệu

- Đọc thư viện thường dùng

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

- Đọc Dataset về rượu vang

- Trong bộ data này, rượu vang được đánh giá, phân loại dựa trên 13 yếu tố.

[ ]:
```
# Đọc dataset từ file csv
df =pd.read_csv('./Data/wine.csv')
```

[ ]:
```
# Khảo sát tổng quan về dữ liệu
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Wine                 178 non-null    int64
 1   Alcohol              178 non-null    float64
 2   Malic.acid           178 non-null    float64
 3   Ash                  178 non-null    float64
 4   Acl                  178 non-null    float64
 5   Mg                   178 non-null    int64
 6   Phenols              178 non-null    float64
 7   Flavanoids           178 non-null    float64
 8   Nonflavanoid.phenols 178 non-null    float64
 9   Proanth              178 non-null    float64
 10  Color.int            178 non-null    float64
 11  Hue                  178 non-null    float64
 12  OD                   178 non-null    float64
 13  Proline              178 non-null    int64
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

[ ]:
```python
# Khảo sát một vài giá trị của bộ dữ liệu
df.head()
```

[ ]:
```
   Wine  Alcohol  Malic.acid   Ash   Acl   Mg  Phenols  Flavanoids  \
0     1    14.23        1.71  2.43  15.6  127     2.80        3.06
1     1    13.20        1.78  2.14  11.2  100     2.65        2.76
2     1    13.16        2.36  2.67  18.6  101     2.80        3.24
3     1    14.37        1.95  2.50  16.8  113     3.85        3.49
4     1    13.24        2.59  2.87  21.0  118     2.80        2.69

   Nonflavanoid.phenols  Proanth  Color.int   Hue    OD  Proline
0                  0.28     2.29       5.64  1.04  3.92     1065
1                  0.26     1.28       4.38  1.05  3.40     1050
2                  0.30     2.81       5.68  1.03  3.17     1185
3                  0.24     2.18       7.80  0.86  3.45     1480
4                  0.39     1.82       4.32  1.04  2.93      735
```

[ ]:
```python
# Khảo sát các loại rượu vang được phân loại
wine_classes = df['Wine'].unique()
print(wine_classes)
```

```
[1 2 3]
```

- Như vậy, bộ dữ liệu phân loại rượu vang thành 3 loại 1, 2, 3

2. Giảm số chiều bằng PCA

- Bước 1. Chuyển và chuẩn hoá dữ liệu

[ ]:
```python
# Các thư viện sử dụng
from sklearn.model_selection import train_test_split    # Tách dữ liệu ra thành␣
 ↪tập Train và test
from sklearn.preprocessing import StandardScaler        # Chuẩn hoá dữ liệu
```

[ ]:
```python
data = df.to_numpy()            #Chuyển data về dạng numpy
X = data[:,1:]
y = data[:,0]
scaler = StandardScaler()
X_std = scaler.fit_transform(X) #Chuẩn hoá dữ liệu

X_train, X_test, y_train, y_test = train_test_split(X_std, y, test_size=0.2,␣
 ↪random_state=42)
# Ta có thể chuẩn hoá cả bộ dữ liệu X sau đó mới tách, hoặc tách ra thành bộ␣
 ↪train và bộ test rồi sau đó mới chuẩn hoá. Kết quả sẽ khác nhau
```

- Bước 2. Tính ma trận covariance
  Ở đây, để cho đơn giản, ta sử dụng thư viện numpy để tính toán

[ ]:
```
cov_mat = np.cov(X_train.T)                    # Chú ý số chiều của X, vì ta␣
↪có 13 features nên ma trận cov phải có cỡ là 13x13
eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)
```

- Bước 3. Xác định các trị riêng và vector riêng

[ ]:
```
# Các trị riêng
print(eigen_vals)
```

```
[4.83276484 2.47147595 1.58630645 1.0054665  0.9052796  0.65983586
 0.57673014 0.11013005 0.36558439 0.16916164 0.30433143 0.25626719
 0.2268225 ]
```

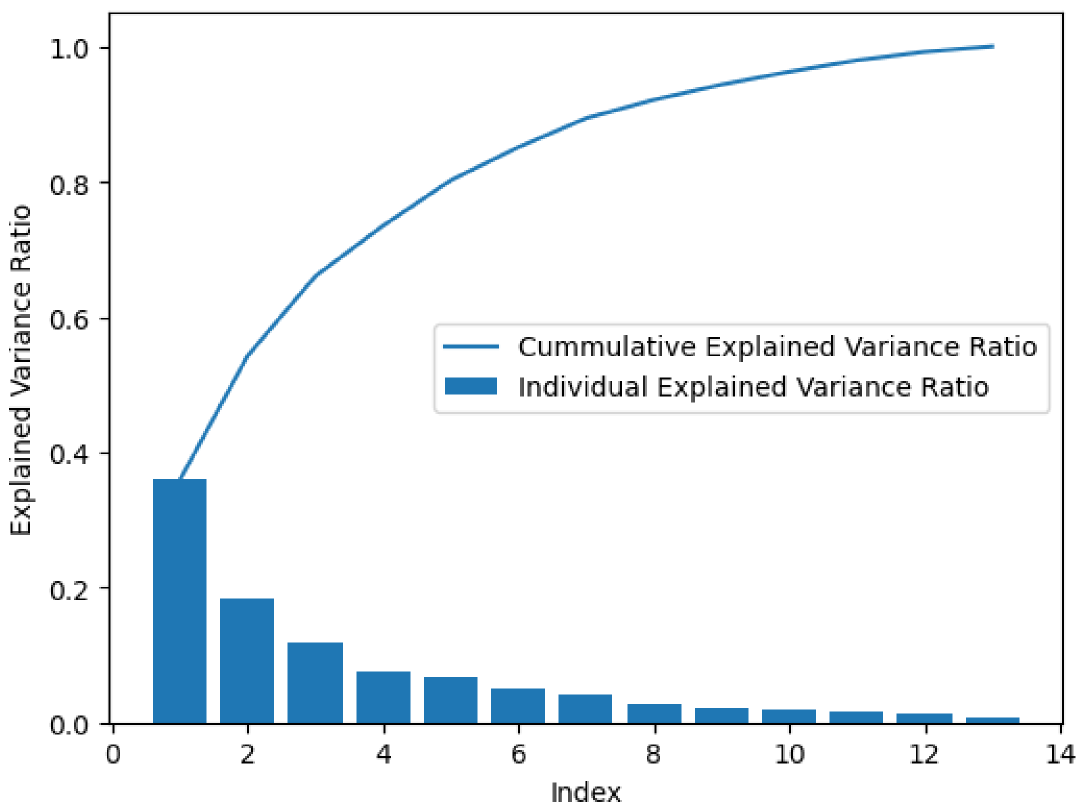
[ ]:
```
# Các vector riêng
print(eigen_vecs)
```

```
[[-0.12066745 -0.49422695  0.19330087 -0.09076028 -0.31323051  0.19428065
  -0.04263048  0.03575323  0.46669715 -0.21479096 -0.4166945   0.29105914
  -0.18528721]
 [ 0.26133943 -0.19247019 -0.13197297 -0.4164102   0.25383008  0.63892782
   0.37636621  0.03427017  0.03121923  0.10376641  0.00693055 -0.24607982
   0.12654151]
 [ 0.02788356 -0.32757004 -0.59903776  0.18904029 -0.20981424  0.13647221
  -0.17676031 -0.17511845 -0.10828485 -0.0790853   0.31088524 -0.15651745
  -0.48337679]
 [ 0.2496346   0.03349307 -0.60443358  0.05586011  0.10238804 -0.12669829
  -0.26586667  0.12986025  0.39270508 -0.04533047 -0.16380853  0.14047044
   0.50117724]
 [-0.14687697 -0.27589848 -0.06119482  0.68395831  0.46239808  0.06725421
   0.32562812  0.06160132 -0.13357867  0.05614285 -0.26698329  0.10046626
   0.01520112]
 [-0.38510351 -0.11126263 -0.1738233  -0.24508968 -0.00468661 -0.08381211
  -0.0786755  -0.40305703 -0.39065152 -0.3033302  -0.43134518 -0.26323998
   0.2708679 ]
 [-0.4148827  -0.0290841  -0.17862017 -0.17940617 -0.02457644 -0.0074331
  -0.08032223  0.8346762  -0.17812762  0.04361835 -0.06183028 -0.11975287
  -0.08569545]
 [ 0.30655131 -0.07092196 -0.20314841 -0.07316583 -0.46865851 -0.32592567
   0.60754939  0.09700533 -0.28381182  0.02628963 -0.13328698  0.22126496
   0.04489239]
 [-0.30951491 -0.08862596 -0.14733562 -0.30769197  0.38266324 -0.44095992
   0.3780716  -0.11203341  0.38184395 -0.1625314   0.2768179   0.0477786
  -0.180181  ]
 [ 0.12396866 -0.52922417  0.13629651 -0.10360611  0.04313467 -0.38229614
  -0.18768689 -0.04014775  0.01421425  0.6289806  -0.05989678 -0.29483045
   0.06373517]
 [-0.30522527  0.28130703 -0.08595284  0.25689253 -0.382161    0.074235
   0.29137453 -0.08257467  0.42237417  0.24694323 -0.09752602 -0.50645827
   0.0751962 ]
```

```
[-0.38583351  0.13120288 -0.21753349 -0.1604521  -0.0346486   0.2129628
 -0.02411567 -0.23612204 -0.07960257  0.58064561 -0.0015121   0.56075444
  0.00547996]
[-0.25469502 -0.36278817  0.13749436  0.1302059  -0.2309406   0.1002935
  0.0562602   0.0306188  -0.04245492 -0.12988368  0.57808154  0.08751365
  0.58279735]]
```

- Bước 4: Sử dụng đồ thị để kiểm tra các giá trị riêng và chọn số chiều

[ ]:
```python
tot = sum(eigen_vals)
sorted_eigen_vals = sorted(eigen_vals, reverse=True)
var_exp = [i/tot for i in sorted_eigen_vals]
cum_var_exp = np.cumsum(var_exp)

plt.bar(range(1,14),var_exp,label = 'Individual Explained Variance Ratio')
plt.plot(range(1,14), cum_var_exp, label = 'Cummulative Explained Variance␣
 ↪Ratio')
plt.legend()
plt.xlabel('Index')
plt.ylabel('Explained Variance Ratio')
plt.show()
```



[ ]:
```
print('Trường hợp giảm số chiều xuống còn 3:', cum_var_exp[2])
print('Trường hợp giảm số chiều xuống còn 2:',cum_var_exp[1])
```

```
Trường hợp giảm số chiều xuống còn 3: 0.6600181085987932
Trường hợp giảm số chiều xuống còn 2: 0.5422535936254841
```

- Ở đây ta thấy: Với 3 giá trị riêng ban đầu, thông tin được giữ lại khoảng 67%, tương ứng, với 2 giá trị riêng là 55%
- Để dễ biểu diễn, ta sẽ rút gọn lại từ 13 chiều (features) còn 2 chiều

[ ]:
```
idx = eigen_vals.argsort()[::-1][:2]  #[:2] chỉ ra lấy 2 giá trị
print(idx)
```

```
[0 1]
```

- Bước 5. Ghép 2 vector riêng tương ứng vào thành ma trận w

[ ]:
```
w = np.zeros((2,eigen_vecs.shape[0]))
for i in range(idx.shape[0]):
    w[i] = eigen_vecs[:,idx[i]]

print(w)
```

```
[[-0.12066745  0.26133943  0.02788356  0.2496346  -0.14687697 -0.38510351
  -0.4148827   0.30655131 -0.30951491  0.12396866 -0.30522527 -0.38583351
  -0.25469502]
 [-0.49422695 -0.19247019 -0.32757004  0.03349307 -0.27589848 -0.11126263
  -0.0290841  -0.07092196 -0.08862596 -0.52922417  0.28130703  0.13120288
  -0.36278817]]
```

- Bước 6. Chuyển dữ liệu sang chiều mới

[ ]:
```
X_new = np.matmul(X_train, w.T)
print(X_new.shape)
print(X_new)
```

```
(142, 2)
[[ 1.28328276 -3.66849844]
 [ 4.00070341 -0.46509052]
 [-2.30831412  1.24498331]
 [ 1.78045004 -2.51051429]
 [-1.55176216  0.74393356]
 [-0.23581474  2.20678463]
 [-0.92588368  2.0013737 ]
 [ 0.77086058  0.25127664]
 [ 2.40174555 -0.32405247]
 [-2.42049794 -1.23297974]
 [-0.95134722  2.42818741]
 [ 1.16022423  0.71173717]
 [ 2.72687019 -0.30117268]
 [-2.78171797 -0.89571315]
```

```
[-0.52225518  2.17632487]
[-2.68529806 -1.59282087]
[-1.70800197 -0.83018416]
[ 2.87690459 -2.10307509]
[ 3.10958103 -0.20801443]
[-0.50582089  2.07494128]
[-1.39335668  0.73870884]
[-1.35170533 -0.73523281]
[-0.8827509   2.26307593]
[-2.53944309 -0.03962867]
[ 4.29870186 -0.3333188 ]
[-1.18070781  2.48379256]
[-2.17548247  1.50761058]
[ 3.00506991 -1.07334682]
[-1.70775122 -0.64994195]
[-2.37263619 -1.3350246 ]
[-1.23394739  0.08611607]
[ 2.23451697 -0.88509363]
[-0.95575942 -0.94877195]
[-1.61914956 -0.17646365]
[-2.62185211 -0.23211254]
[ 3.1660967   0.49583298]
[ 0.69597364  2.57180109]
[-3.37622777 -1.44935112]
[ 3.66818174 -1.07158616]
[ 0.43824024  0.63158374]
[-3.26931851 -1.62382706]
[ 3.01385955 -0.2926403 ]
[-0.09478127  1.13413419]
[ 0.49295749  1.96662767]
[-2.12050446 -0.06847357]
[ 0.45566768  0.61149444]
[-2.12491314 -0.85041325]
[-2.56418778 -0.99391173]
[ 1.58452764  1.17187441]
[-0.91242721 -1.04444439]
[-1.65982342  0.44663397]
[ 2.93134902  0.03200063]
[-1.10217933  1.40459138]
[-0.41263802  0.96470003]
[-0.93068276  1.37584596]
[-2.48200209 -1.88910661]
[ 0.39162119  2.38333348]
[ 0.69882411  2.27860776]
[-0.80661586  1.31057114]
[ 3.50106015 -2.01059411]
[ 2.87224295 -0.41841926]
[-2.789616   -0.82079026]
```

```
[-1.09711071  2.2288408 ]
[ 3.07704804 -0.24740998]
[ 2.55252729 -2.40705362]
[ 1.75876229  0.91654557]
[-2.54311881 -0.27194942]
[-1.80986552 -1.73664851]
[-0.01151328  2.05074908]
[ 2.13449305 -0.04380951]
[ 2.4003609  -0.17377225]
[-0.12442534  1.34056868]
[ 2.48102946 -2.163709  ]
[ 1.28573847  0.78762081]
[-0.65397579  3.75430333]
[ 1.68184384  1.66444967]
[-2.09998251 -2.00515452]
[-2.91511959 -2.31441555]
[ 3.19767905 -0.61534652]
[-3.58218167 -2.99429767]
[ 3.58212494 -0.67923691]
[-2.58704517 -1.91582554]
[-1.87765321 -0.28085219]
[-0.95430113  3.33002841]
[ 1.33434918  1.53034617]
[-1.36809751 -0.71881311]
[-2.65019113 -1.35403373]
[-1.97969577 -1.64815786]
[-1.0322437   1.3073142 ]
[-0.45566874 -0.34811172]
[ 1.49557792  1.05819256]
[ 3.93676647  0.10027684]
[-1.5223935   1.28716613]
[ 1.47414412  1.86752505]
[ 2.56026512  0.13013812]
[ 2.99625974 -1.82266626]
[ 2.29010957 -1.00845939]
[ 0.45260468  2.61249773]
[-2.4639186  -1.08198367]
[-3.36331585 -1.48465753]
[ 0.67805987  3.23797096]
[ 1.87317596 -1.36518331]
[ 2.36632145 -0.08488339]
[-1.78429965 -1.7569278 ]
[ 0.66721893  1.15918496]
[ 3.72283469 -1.65676509]
[ 2.34321118  0.4677638 ]
[ 2.89741746 -1.43674056]
[ 2.59724146 -0.46948199]
[-2.14093053  1.78011925]
```

```
 [-2.12235216 -1.05931024]
 [ 1.55441128  1.37926305]
 [-3.19546488 -0.36312636]
 [ 2.79528269 -2.52411946]
 [-3.02765659 -1.90274236]
 [-1.92220263 -1.39104364]
 [ 1.04519763  1.80627096]
 [-1.06519747 -0.27111924]
 [-2.07200398 -1.34016239]
 [ 2.5867415  -2.08680719]
 [ 1.75808317  1.22693132]
 [-1.0979696  -0.13433441]
 [ 3.50629574 -1.18398242]
 [ 2.30116828 -1.9222402 ]
 [-2.24988905  0.24443816]
 [-3.40379443 -1.8711746 ]
 [ 2.98073436 -1.77073341]
 [ 1.25826863  0.08207507]
 [ 2.62872253 -1.79831061]
 [ 0.49775092  2.68919859]
 [-1.50119124  1.95954146]
 [ 0.08353893  2.84058248]
 [ 0.46896043  2.4093518 ]
 [-1.9025232   1.24630744]
 [-1.26071734 -0.46343153]
 [ 3.38757293 -2.66222796]
 [-3.12355814 -0.93996058]
 [-1.66784744  0.85053595]
 [ 0.24966718  2.1491447 ]
 [-4.16412194 -2.40659518]
 [ 1.85892284  1.55325025]
 [ 0.08829371  1.14899623]]
```

[ ]:
```python
X_new = np.matmul(w, X_train.T).T
print(X_new.shape)
print(X_new)
```

```
(142, 2)
[[ 1.28328276 -3.66849844]
 [ 4.00070341 -0.46509052]
 [-2.30831412  1.24498331]
 [ 1.78045004 -2.51051429]
 [-1.55176216  0.74393356]
 [-0.23581474  2.20678463]
 [-0.92588368  2.0013737 ]
 [ 0.77086058  0.25127664]
 [ 2.40174555 -0.32405247]
 [-2.42049794 -1.23297974]
```

```
[-0.95134722  2.42818741]
[ 1.16022423  0.71173717]
[ 2.72687019 -0.30117268]
[-2.78171797 -0.89571315]
[-0.52225518  2.17632487]
[-2.68529806 -1.59282087]
[-1.70800197 -0.83018416]
[ 2.87690459 -2.10307509]
[ 3.10958103 -0.20801443]
[-0.50582089  2.07494128]
[-1.39335668  0.73870884]
[-1.35170533 -0.73523281]
[-0.8827509   2.26307593]
[-2.53944309 -0.03962867]
[ 4.29870186 -0.3333188 ]
[-1.18070781  2.48379256]
[-2.17548247  1.50761058]
[ 3.00506991 -1.07334682]
[-1.70775122 -0.64994195]
[-2.37263619 -1.3350246 ]
[-1.23394739  0.08611607]
[ 2.23451697 -0.88509363]
[-0.95575942 -0.94877195]
[-1.61914956 -0.17646365]
[-2.62185211 -0.23211254]
[ 3.1660967   0.49583298]
[ 0.69597364  2.57180109]
[-3.37622777 -1.44935112]
[ 3.66818174 -1.07158616]
[ 0.43824024  0.63158374]
[-3.26931851 -1.62382706]
[ 3.01385955 -0.2926403 ]
[-0.09478127  1.13413419]
[ 0.49295749  1.96662767]
[-2.12050446 -0.06847357]
[ 0.45566768  0.61149444]
[-2.12491314 -0.85041325]
[-2.56418778 -0.99391173]
[ 1.58452764  1.17187441]
[-0.91242721 -1.04444439]
[-1.65982342  0.44663397]
[ 2.93134902  0.03200063]
[-1.10217933  1.40459138]
[-0.41263802  0.96470003]
[-0.93068276  1.37584596]
[-2.48200209 -1.88910661]
[ 0.39162119  2.38333348]
[ 0.69882411  2.27860776]
```

```
[-0.80661586  1.31057114]
[ 3.50106015 -2.01059411]
[ 2.87224295 -0.41841926]
[-2.789616   -0.82079026]
[-1.09711071  2.2288408 ]
[ 3.07704804 -0.24740998]
[ 2.55252729 -2.40705362]
[ 1.75876229  0.91654557]
[-2.54311881 -0.27194942]
[-1.80986552 -1.73664851]
[-0.01151328  2.05074908]
[ 2.13449305 -0.04380951]
[ 2.4003609  -0.17377225]
[-0.12442534  1.34056868]
[ 2.48102946 -2.163709  ]
[ 1.28573847  0.78762081]
[-0.65397579  3.75430333]
[ 1.68184384  1.66444967]
[-2.09998251 -2.00515452]
[-2.91511959 -2.31441555]
[ 3.19767905 -0.61534652]
[-3.58218167 -2.99429767]
[ 3.58212494 -0.67923691]
[-2.58704517 -1.91582554]
[-1.87765321 -0.28085219]
[-0.95430113  3.33002841]
[ 1.33434918  1.53034617]
[-1.36809751 -0.71881311]
[-2.65019113 -1.35403373]
[-1.97969577 -1.64815786]
[-1.0322437   1.3073142 ]
[-0.45566874 -0.34811172]
[ 1.49557792  1.05819256]
[ 3.93676647  0.10027684]
[-1.5223935   1.28716613]
[ 1.47414412  1.86752505]
[ 2.56026512  0.13013812]
[ 2.99625974 -1.82266626]
[ 2.29010957 -1.00845939]
[ 0.45260468  2.61249773]
[-2.4639186  -1.08198367]
[-3.36331585 -1.48465753]
[ 0.67805987  3.23797096]
[ 1.87317596 -1.36518331]
[ 2.36632145 -0.08488339]
[-1.78429965 -1.7569278 ]
[ 0.66721893  1.15918496]
[ 3.72283469 -1.65676509]
```

```
 [ 2.34321118  0.4677638 ]
 [ 2.89741746 -1.43674056]
 [ 2.59724146 -0.46948199]
 [-2.14093053  1.78011925]
 [-2.12235216 -1.05931024]
 [ 1.55441128  1.37926305]
 [-3.19546488 -0.36312636]
 [ 2.79528269 -2.52411946]
 [-3.02765659 -1.90274236]
 [-1.92220263 -1.39104364]
 [ 1.04519763  1.80627096]
 [-1.06519747 -0.27111924]
 [-2.07200398 -1.34016239]
 [ 2.5867415  -2.08680719]
 [ 1.75808317  1.22693132]
 [-1.0979696  -0.13433441]
 [ 3.50629574 -1.18398242]
 [ 2.30116828 -1.9222402 ]
 [-2.24988905  0.24443816]
 [-3.40379443 -1.8711746 ]
 [ 2.98073436 -1.77073341]
 [ 1.25826863  0.08207507]
 [ 2.62872253 -1.79831061]
 [ 0.49775092  2.68919859]
 [-1.50119124  1.95954146]
 [ 0.08353893  2.84058248]
 [ 0.46896043  2.4093518 ]
 [-1.9025232   1.24630744]
 [-1.26071734 -0.46343153]
 [ 3.38757293 -2.66222796]
 [-3.12355814 -0.93996058]
 [-1.66784744  0.85053595]
 [ 0.24966718  2.1491447 ]
 [-4.16412194 -2.40659518]
 [ 1.85892284  1.55325025]
 [ 0.08829371  1.14899623]]
```

3. Sử dụng PCA bằng scikit-learn

[ ]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_train_new_2 = pca.fit_transform(X_train)
```

[ ]:
```
print(X_train_new_2.shape)
print(X_train_new_2)
```

```
(142, 2)
[[ 1.23838431 -3.67303077]
 [ 3.95580496 -0.46962285]
```

```
[-2.35321257  1.24045098]
[ 1.73555159 -2.51504663]
[-1.59666061  0.73940123]
[-0.28071319  2.20225229]
[-0.97078213  1.99684136]
[ 0.72596213  0.2467443 ]
[ 2.35684711 -0.3285848 ]
[-2.46539639 -1.23751208]
[-0.99624567  2.42365507]
[ 1.11532579  0.70720483]
[ 2.68197174 -0.30570501]
[-2.82661642 -0.90024549]
[-0.56715363  2.17179254]
[-2.73019651 -1.5973532 ]
[-1.75290042 -0.8347165 ]
[ 2.83200614 -2.10760742]
[ 3.06468258 -0.21254676]
[-0.55071934  2.07040895]
[-1.43825513  0.7341765 ]
[-1.39660377 -0.73976514]
[-0.92764934  2.2585436 ]
[-2.58434154 -0.04416101]
[ 4.25380341 -0.33785114]
[-1.22560626  2.47926023]
[-2.22038092  1.50307824]
[ 2.96017146 -1.07787915]
[-1.75264967 -0.65447428]
[-2.41753463 -1.33955693]
[-1.27884583  0.08158373]
[ 2.18961853 -0.88962596]
[-1.00065787 -0.95330429]
[-1.66404801 -0.18099598]
[-2.66675056 -0.23664488]
[ 3.12119825  0.49130064]
[ 0.65107519  2.56726876]
[-3.42112621 -1.45388345]
[ 3.62328329 -1.0761185 ]
[ 0.3933418   0.62705141]
[-3.31421696 -1.6283594 ]
[ 2.9689611  -0.29717264]
[-0.13967972  1.12960186]
[ 0.44805904  1.96209534]
[-2.1654029  -0.07300591]
[ 0.41076923  0.6069621 ]
[-2.16981159 -0.85494559]
[-2.60908623 -0.99844406]
[ 1.5396292   1.16734208]
[-0.95732566 -1.04897672]
```

```
[-1.70472186  0.44210164]
[ 2.88645057  0.02746829]
[-1.14707778  1.40005904]
[-0.45753646  0.9601677 ]
[-0.97558121  1.37131363]
[-2.52690054 -1.89363895]
[ 0.34672275  2.37880115]
[ 0.65392567  2.27407543]
[-0.8515143   1.3060388 ]
[ 3.45616171 -2.01512644]
[ 2.8273445  -0.42295159]
[-2.83451444 -0.82532259]
[-1.14200916  2.22430847]
[ 3.03214959 -0.25194231]
[ 2.50762884 -2.41158595]
[ 1.71386384  0.91201323]
[-2.58801726 -0.27648175]
[-1.85476397 -1.74118084]
[-0.05641172  2.04621675]
[ 2.0895946  -0.04834185]
[ 2.35546245 -0.17830459]
[-0.16932379  1.33603635]
[ 2.43613101 -2.16824133]
[ 1.24084002  0.78308847]
[-0.69887424  3.749771  ]
[ 1.63694539  1.65991734]
[-2.14488096 -2.00968686]
[-2.96001804 -2.31894789]
[ 3.1527806  -0.61987886]
[-3.62708011 -2.99883001]
[ 3.53722649 -0.68376925]
[-2.63194362 -1.92035788]
[-1.92255166 -0.28538452]
[-0.99919957  3.32549608]
[ 1.28945074  1.52581384]
[-1.41299596 -0.72334544]
[-2.69508958 -1.35856606]
[-2.02459422 -1.6526902 ]
[-1.07714215  1.30278186]
[-0.50056719 -0.35264405]
[ 1.45067947  1.05366023]
[ 3.89186802  0.09574451]
[-1.56729195  1.28263379]
[ 1.42924568  1.86299271]
[ 2.51536667  0.12560579]
[ 2.95136129 -1.82719859]
[ 2.24521112 -1.01299172]
[ 0.40770623  2.60796539]
```

```
 [-2.50881705 -1.086516  ]
 [-3.4082143  -1.48918987]
 [ 0.63316143  3.23343862]
 [ 1.82827751 -1.36971565]
 [ 2.321423   -0.08941573]
 [-1.8291981  -1.76146014]
 [ 0.62232048  1.15465263]
 [ 3.67793624 -1.66129742]
 [ 2.29831273  0.46323147]
 [ 2.85251901 -1.4412729 ]
 [ 2.55234301 -0.47401432]
 [-2.18582897  1.77558692]
 [-2.16725061 -1.06384258]
 [ 1.50951283  1.37473072]
 [-3.24036333 -0.36765869]
 [ 2.75038425 -2.52865179]
 [-3.07255503 -1.90727469]
 [-1.96710108 -1.39557597]
 [ 1.00029919  1.80173862]
 [-1.11009591 -0.27565158]
 [-2.11690243 -1.34469473]
 [ 2.54184306 -2.09133952]
 [ 1.71318472  1.22239899]
 [-1.14286805 -0.13886674]
 [ 3.4613973  -1.18851475]
 [ 2.25626983 -1.92677254]
 [-2.29478749  0.23990582]
 [-3.44869288 -1.87570694]
 [ 2.93583591 -1.77526575]
 [ 1.21337018  0.07754273]
 [ 2.58382408 -1.80284294]
 [ 0.45285247  2.68466626]
 [-1.54608968  1.95500912]
 [ 0.03864048  2.83605015]
 [ 0.42406198  2.40481946]
 [-1.94742165  1.2417751 ]
 [-1.30561579 -0.46796386]
 [ 3.34267448 -2.66676029]
 [-3.16845659 -0.94449291]
 [-1.71274589  0.84600362]
 [ 0.20476873  2.14461236]
 [-4.20902039 -2.41112752]
 [ 1.81402439  1.54871791]
 [ 0.04339527  1.14446389]]
```

[ ]:

```
pca.components_
```

[ ]: array([[-0.12066745, 0.26133943, 0.02788356, 0.2496346 , -0.14687697,
        -0.38510351, -0.4148827 , 0.30655131, -0.30951491, 0.12396866,
        -0.30522527, -0.38583351, -0.25469502],
       [-0.49422695, -0.19247019, -0.32757004, 0.03349307, -0.27589848,
        -0.11126263, -0.0290841 , -0.07092196, -0.08862596, -0.52922417,
         0.28130703, 0.13120288, -0.36278817]])

[ ]:

4. Thực hiện bài toán phân loại như bình thường

[ ]:
```
from matplotlib.colors import ListedColormap
def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):
    # setup marker generator and color map
    markers = ('o', 's', '^', 'v', '<')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
    np.arange(x2_min, x2_max, resolution))
    lab = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    lab = lab.reshape(xx1.shape)
    plt.contourf(xx1, xx2, lab, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    # plot class examples



    classes = np.unique(y)
    print(X.shape)
    for idx in range(classes.shape[0]):
        plt.scatter(X[ y==classes[idx],0], y=X[ y == classes[idx],1],␣
 ↪marker=markers[idx])
```

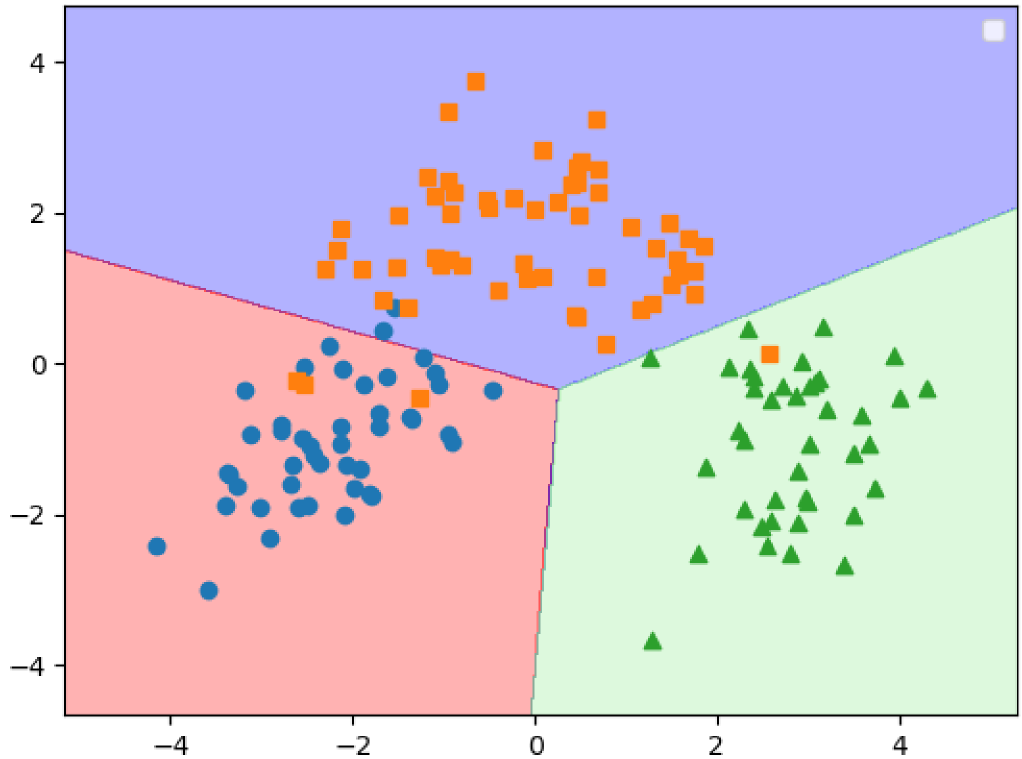
[ ]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(multi_class='ovr', random_state=1, solver='lbfgs')
lr.fit(X_new, y_train)
```

[ ]: LogisticRegression(multi_class='ovr', random_state=1)

[ ]:
```
plot_decision_regions(X_new, y_train, classifier=lr)
```

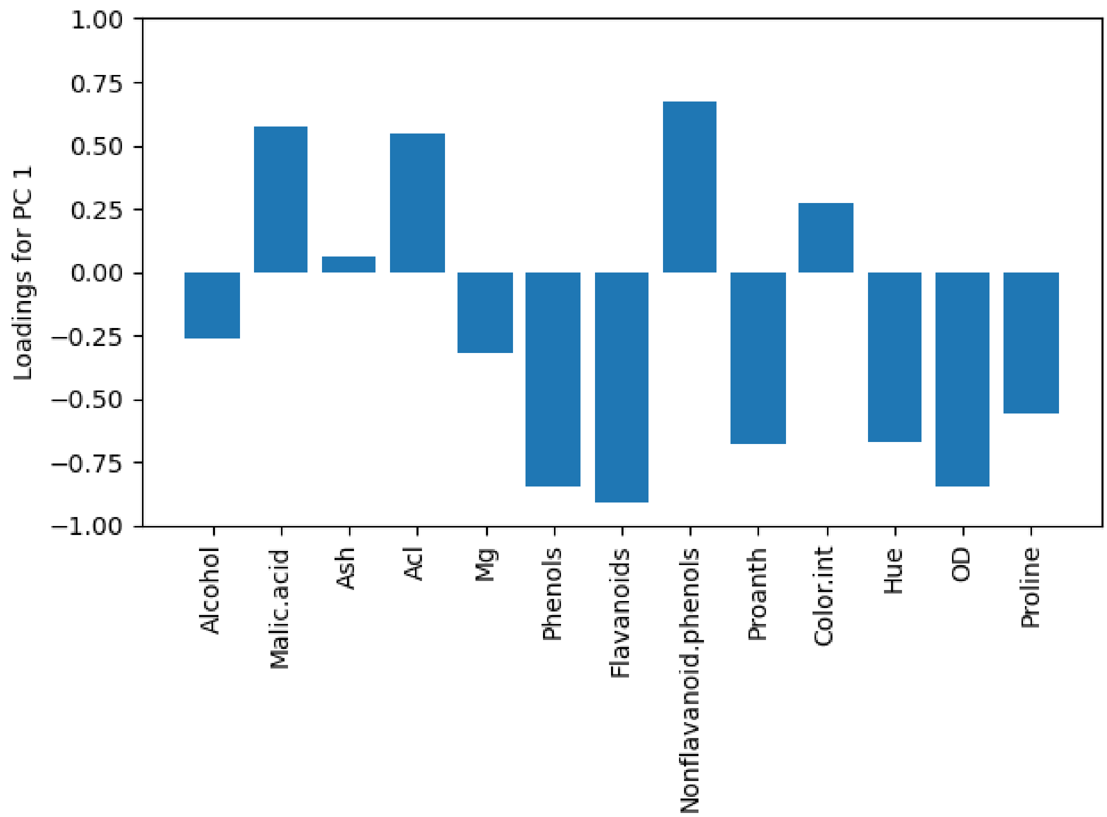
No artists with labels found to put in legend. Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.

(142, 2)



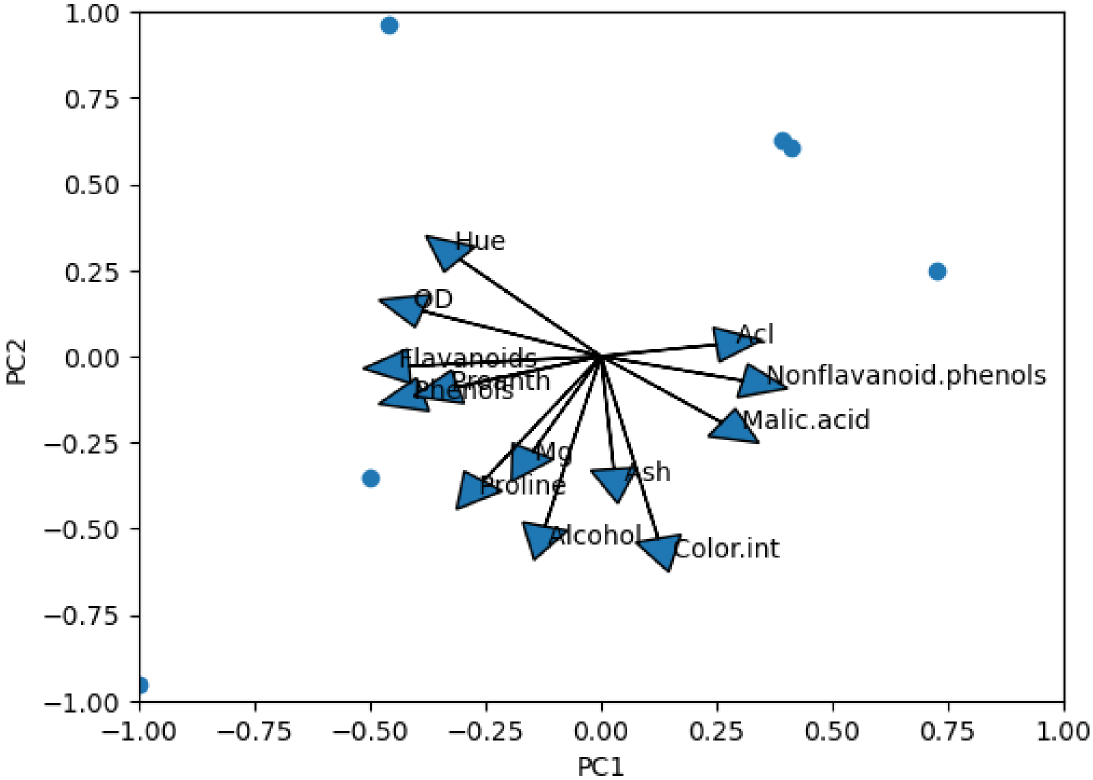
Thông tin về mỗi feature chứa trong thành phần chính

[ ]:
```
loadings = eigen_vecs * np.sqrt(eigen_vals)
fig, ax = plt.subplots()
ax.bar(range(13), loadings[:, 0], align='center')
ax.set_ylabel('Loadings for PC 1')
ax.set_xticks(range(13))
ax.set_xticklabels(df.columns[1:], rotation=90)
plt.ylim([-1, 1])
plt.tight_layout()
plt.show()
```



[ ]:
```python
loadings = pca.components_
scores = X_train_new_2
fig, ax = plt.subplots()

# Plot the scores
ax.scatter(scores[:, 0], scores[:, 1])

# Plot the loadings
for i, v in enumerate(loadings.T):
    ax.arrow(0, 0, v[0], v[1], head_width=0.1, head_length=0.1)
    ax.text(v[0] * 1.1, v[1] * 1.1, f" {df.columns[i+1]}")

# Set the axis limits
ax.set_xlim(-1, 1)
ax.set_ylim(-1, 1)

# Set the axis labels
ax.set_xlabel("PC1")
ax.set_ylabel("PC2")
```

```python
# Show the plot
plt.show()
```




```
[ ]:
```